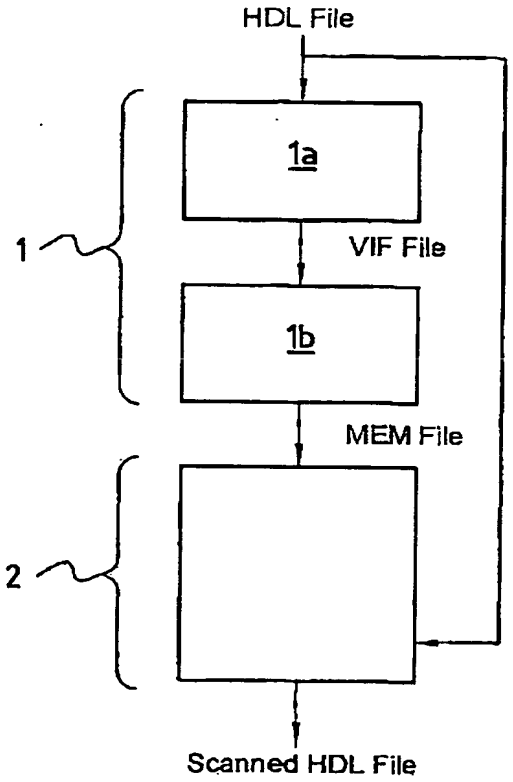


FIG.1

MEM File

```
S_O grant_o CLOCK 35 std_logic_vector (3:0) b03 BEHAV /signal-variable nom
horloge-synchronisation type taille nom-entité nom-architecture /
VAR coda0 CLOCK 35 std_logic_vector (2:0) b03 BEHAV
VAR coda1 CLOCK 35 std_logic_vector (2:0) b03 BEHAV
VAR coda2 CLOCK 35 std_logic_vector (2:0) b03 BEHAV
VAR coda3 CLOCK 35 std_logic_vector (2:0) b03 BEHAV
VAR fu1 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR fu2 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR fu3 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR fu4 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR grant CLOCK 35 std_logic_vector (3:0) b03 BEHAV
VAR ru1 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR ru2 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR ru3 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR ru4 CLOCK 35 std_logic (0:0) b03 BEHAV
VAR stato CLOCK 35 std_logic_vector (1:0) b03 BEHAV
PROCESS 1
```

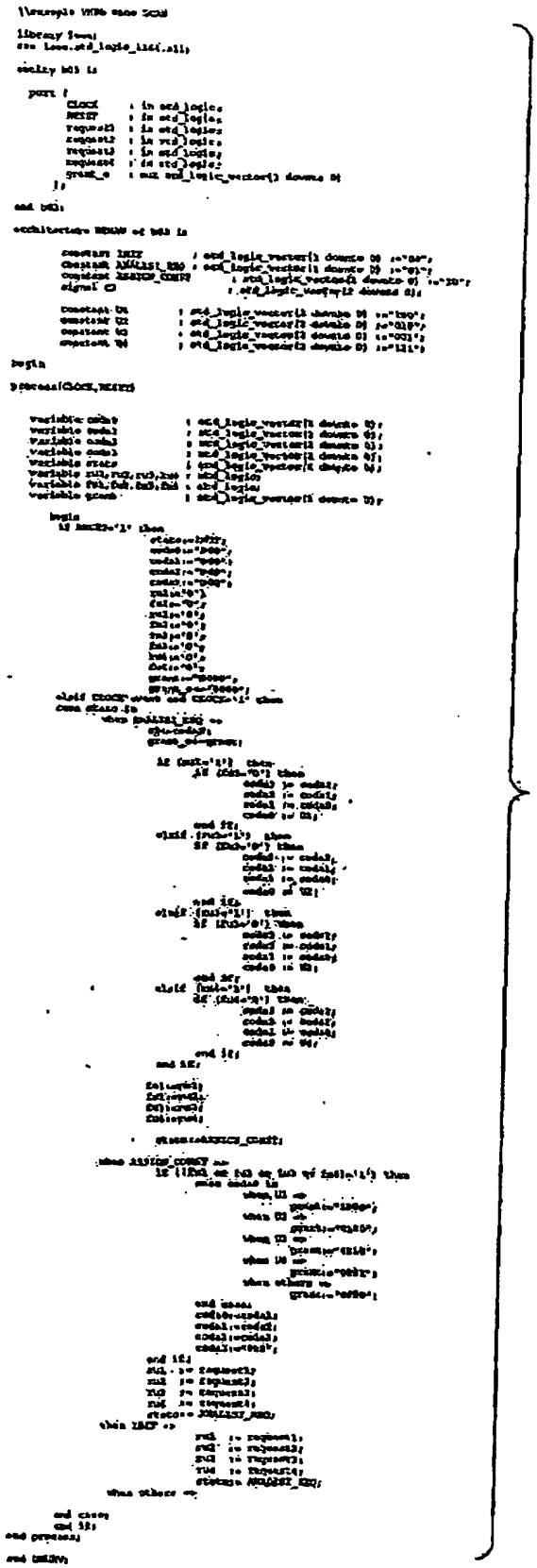
FIG.6

MEM File

```
S_O A_Q_OUT CLOCK 20 REG (7:0) example_4_processes
S_O B_Q_OUT CLOCK 26 REG (7:0) example_4_processes
S_O C_Q_OUT CLOCK 35 REG (7:0) example_4_processes
S_O D_Q_OUT CLOCK 44 REG (7:0) example_4_processes
PROCESS 4
EOF
```

FIG.7

FIG.2



HDL File

VIF File

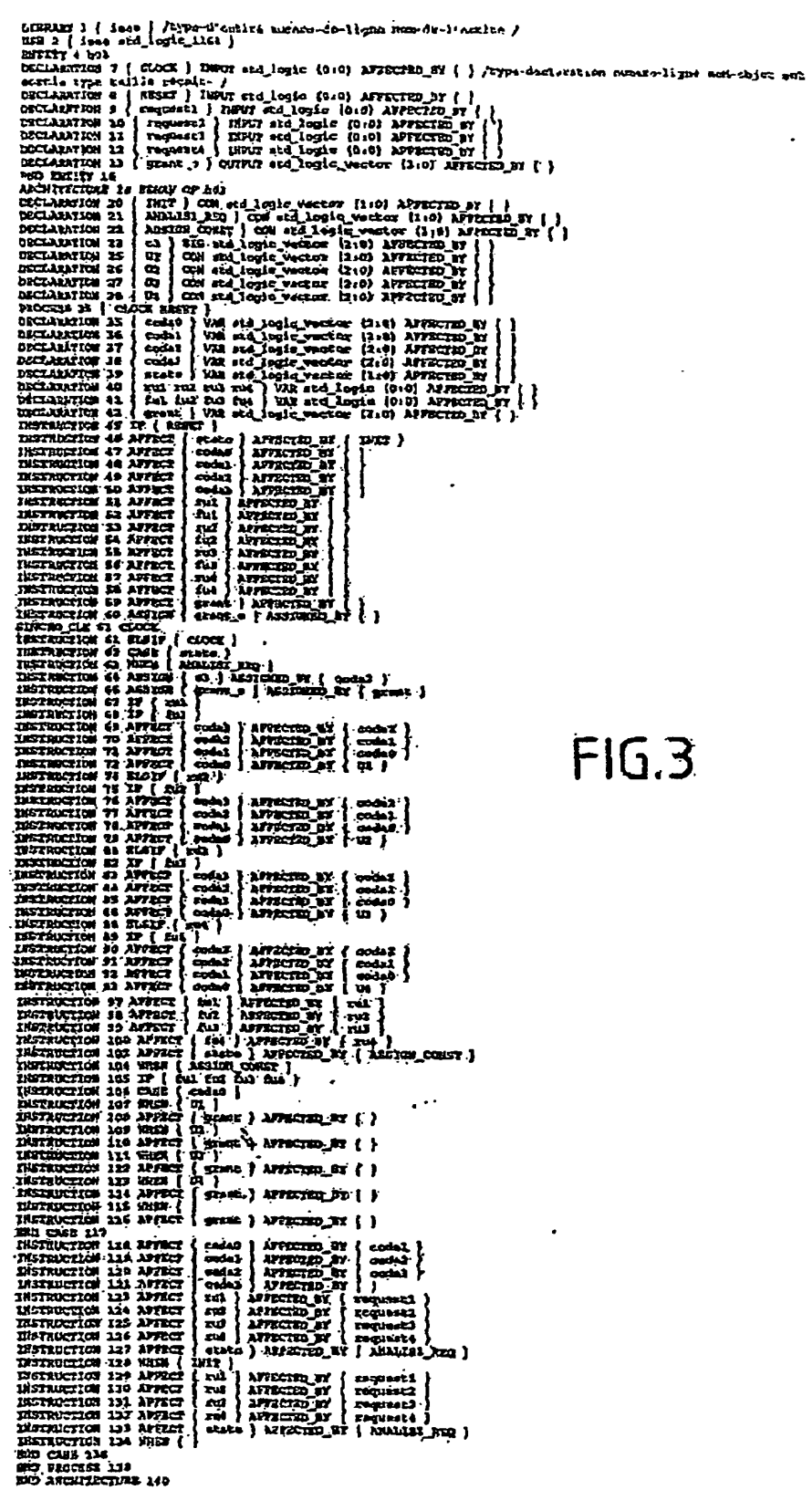


FIG.3

```
// Example of Multiple Processes Verilog sans scan


module example_4_processes (RESET, CLOCK, ENABLE, D_IN,
                  A_Q_OUT, B_Q_OUT, C_Q_OUT, D_Q_OUT);

input RESET, CLOCK, ENABLE;
input       [7:0] D_IN;
output      [7:0] A_Q_OUT;
output      [7:0] B_Q_OUT;
output      [7:0] C_Q_OUT;
output      [7:0] D_Q_OUT;

reg         [7:0] A_Q_OUT;
reg         [7:0] B_Q_OUT;
reg         [7:0] C_Q_OUT;
reg         [7:0] D_Q_OUT;

    // D flip-flop
    always @(posedge CLOCK)
    begin
        A_Q_OUT = D_IN;
    end

    // Flip-flop with asynchronous reset
    always @(posedge CLOCK)
    begin
        if (RESET)
            B_Q_OUT = 8'b00000000;
            else
               B_Q_OUT = D_IN;
    end

    // Flip-flop with asynchronous set
    always @(posedge CLOCK)
    begin
        if (RESET)
            C_Q_OUT = 8'b11111111;
            else
                C_Q_OUT = D_IN;
    end

    //Flip-flop with asynchronous reset & clock enable
    always @(posedge CLOCK)
    begin
            if (RESET)
                 D_Q_OUT = 8'b00000000;
            else if (ENABLE)
                 D_Q_OUT = D_IN;
    end

endmodule
```

HDL File

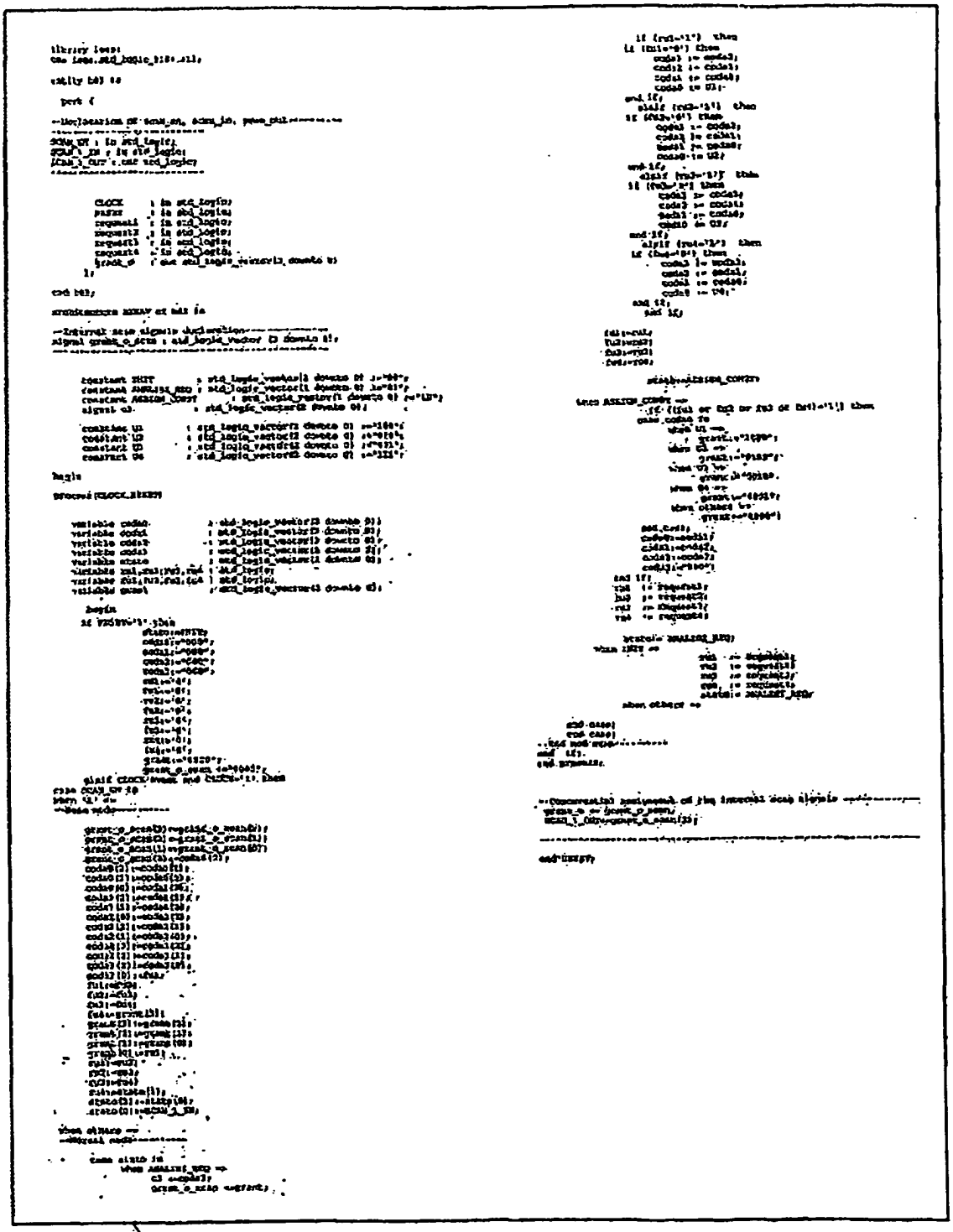
FIG.4

EOP

VIF File

```
MODULE 5 example_4_processes { A_Q_OUT B_Q_OUT CLOCK C_Q_OUT D_IN D_Q_OUT ENABLE
RESET }
DECLARATION 7 INPUT $0:0$ { CLOCK ENABLE RESET }
DECLARATION 8 INPUT $7:0$ { D_IN }
DECLARATION 9 OUTPUT $7:0$ { A_Q_OUT }
DECLARATION 10 OUTPUT $7:0$ { B_Q_OUT }
DECLARATION 11 OUTPUT $7:0$ { C_Q_OUT }
DECLARATION 12 OUTPUT $7:0$ { D_Q_OUT }
DECLARATION 14  REG $7:0$ { A_Q_OUT }
DECLARATION 15  REG $7:0$ { B_Q_OUT }
DECLARATION 16  REG $7:0$ { C_Q_OUT }
DECLARATION 17  REG $7:0$ { D_Q_OUT }
PROCESS 20 { CLOCK }
SINCRO_CLK 20 { CLOCK }
INSTRUCTION 22 AFFECT { A_Q_OUT } AFFECTED_BY { D_IN }
END PROCESS 23
PROCESS 26 { CLOCK }
SINCRO_CLK 26 { CLOCK }
BEGIN_SECV 28 CLOCK
INSTRUCTION 28 IF { RESET }
INSTRUCTION 29 AFFECT { B_Q_OUT } AFFECTED_BY { }
INSTRUCTION 30 ELSE
INSTRUCTION 31 AFFECT { B_Q_OUT } AFFECTED_BY { D_IN }
END PROCESS 32
PROCESS 35 { CLOCK }
SINCRO_CLK 35 { CLOCK }
BEGIN_SECV 37 CLOCK
INSTRUCTION 37 IF { RESET }
INSTRUCTION 38 AFFECT { C_Q_OUT } AFFECTED_BY { }
INSTRUCTION 39 ELSE
INSTRUCTION 40 AFFECT { C_Q_OUT } AFFECTED_BY { D_IN }
END PROCESS 41
PROCESS 44 { CLOCK }
SINCRO_CLK 44 { CLOCK }
BEGIN_SECV 46 CLOCK
INSTRUCTION 46 IF { RESET }
INSTRUCTION 47 AFFECT { D_Q_OUT } AFFECTED_BY { }
INSTRUCTION 48 ELSE
INSTRUCTION 48 IF { ENABLE }
INSTRUCTION 49 AFFECT { D_Q_OUT } AFFECTED_BY { D_IN }
END PROCESS 50
ENDMODULE 52 example_4_processes
```

FIG.5



Scanned HDL File

FIG.8

Scanned HDL File

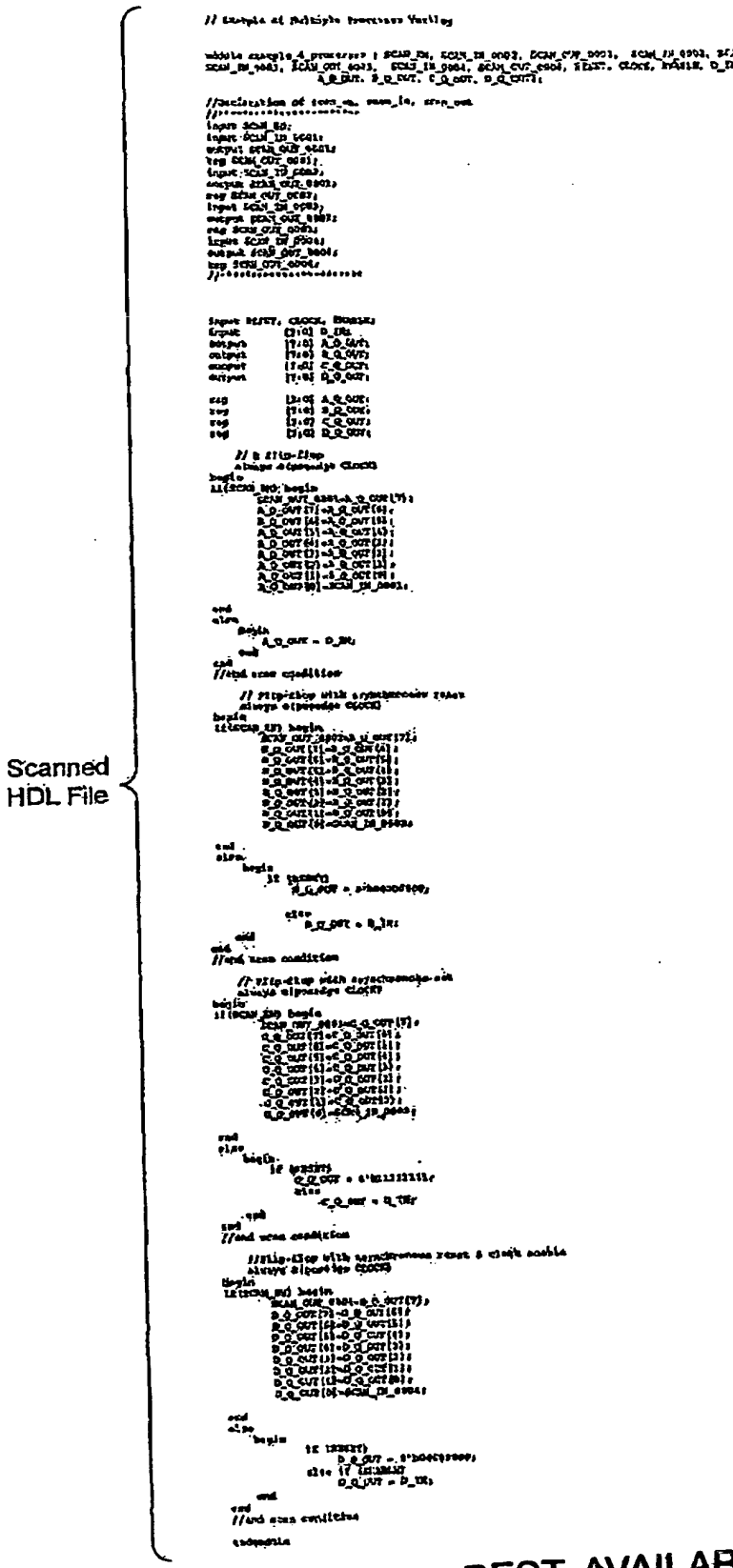


FIG.9

FIG.10

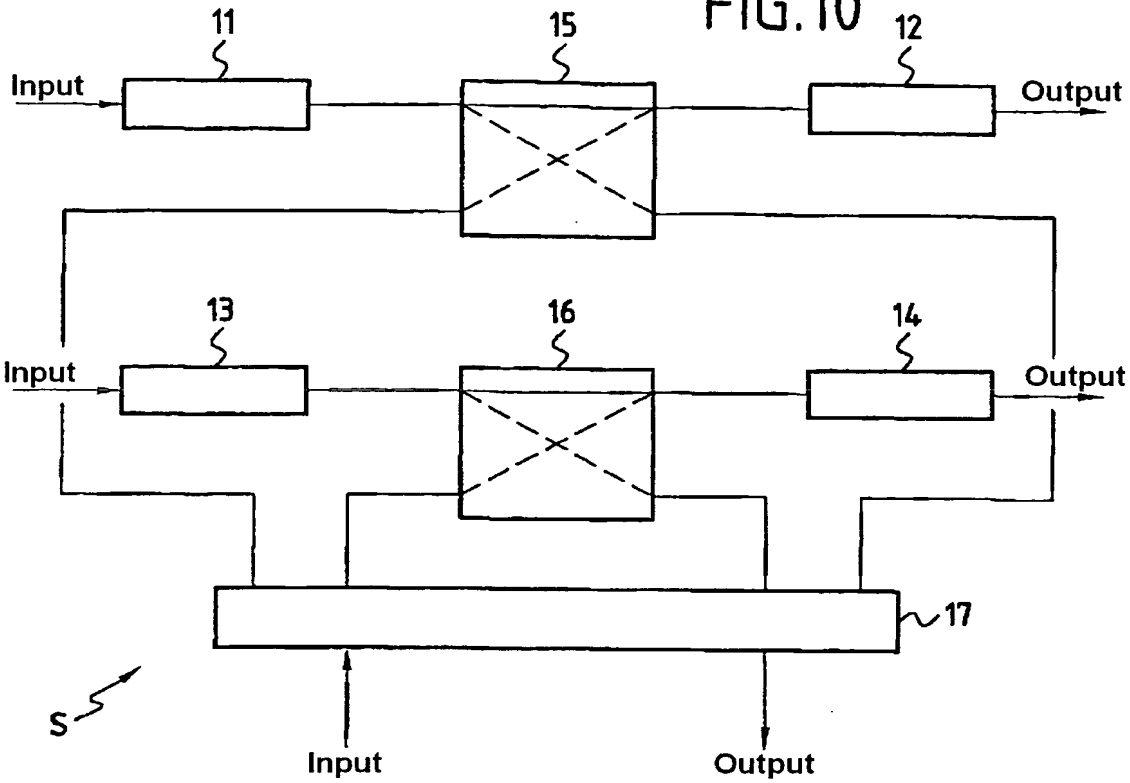


FIG.11